PENYEIMBANGAN BEBAN *TRANSPARENT* SQUID/LUSCA *PROXY* DENGAN METODE *DESTINATION NAT ROUND ROBIN* DENGAN *MULTIPLE CAPTIVE PORTAL* SEBAGAI MEDIA AUTENTIKASI UNTUK *VLAN* TERPADU

Yosua Alvin Adi Soetrisno[*)], Adian Fatchur Rochim, and R. Rizal Isnanto

Jurusan Teknik Elektro, Fakultas Teknik, Universitas Diponegoro,
Jln. Prof. Sudharto, Tembalang, Semarang, Indonesia

[*)]*E-mail: yosuaalvin@yahoo.com*

Abstrak

Pengunaan media Internet pada lingkungan kampus semakin hari semakin meningkat. Peningkatan jumlah pengguna ini bisa menjadi tidak sebanding dengan ketersediaan bandwidth pada sistem yang ada. Masalah yang timbul adalah bahwa penggunaan server proxy tunggal, seiring dengan meningkatnya pengguna, dapat menyebabkan overload dan bisa berdampak pada menurunnya kinerja jaringan. Masalah ini dapat diatasi dengan melakukan penyeimbangan beban pada beberapa server proxy, sehingga dapat mengurangi resiko overload. Masalah lain yang timbul adalah bahwa server proxy secara transparan tidak bisa melakukan proses autentikasi sehingga perlu digunakan aplikasi khusus untuk menangani proses tersebut. Squid merupakan sebuah aplikasi proxy yang telah banyak digunakan pada taraf produksi, sehingga akan digunakan pada penelitian ini. LUSCA akan digunakan sebagai head dari Squid sehingga bisa melakukan caching konten dinamis. Linux akan menyediakan sebuah firewall bawaan bernama IPTables yang bisa digunakan untuk melakukan penyeimbangan beban. Penyeimbangan beban pada IPTables akan menggunakan algoritma round robin. Round robin akan mendistribusikan semua beban secara bergantian ke beberapa server proxy yang ada, dengan menggunakan sistem antrian sirkuler. Aplikasi khusus yang digunakan untuk membantu proses autentikasi adalah captive portal. Dari hasil pengujian, captive portal bisa digunakan untuk menangani autentikasi pengguna dengan menggunakan username dan password yang terdaftar di server LDAP. Pengujian lain yang dilakukan adalah pengujian waktu respon untuk memuat halaman web sebelum dan sesudah menggunakan server proxy. Penggunaan server proxy dapat mempercepat waktu respon untuk memuat halaman web. Kinerja server proxy dapat ditinjau dari persentase rasio hit yang terjadi. Penyeimbangan beban dapat meningkatkan rasio hit dari masing-masing server, berkisar antara 20-30%.

Kata Kunci: proxy, caching, transparan, autentikasi, RADIUS, captive portal, penyeimbangan beban, IPTables, round robin, rasio hit

Abstract

The use of Internet media in a campus environment is increasingly rising. Increase in the number of users this could be not worth the bandwidth available on existing systems. The problem that arises is that the use of a single proxy server, along with the increase of users, can lead to overload and can decrease the network performance problem can be solved by balancing the load on a proxy server, so can reduce the risk of overload. Squid is a proxy application that has been widely used in the production stage, so it will be used in this study. LUSCA will be used as the head of the Squid so that it can do caching of dynamic content. Linux will provide a default firewall called IPTables that can be used to perform load balancing. Balancing the load on IPTables will use a round robin algorithm. Round robin will distribute all loads interchangeably to some existing proxy server, using a circular queue. Specific applications that are used to help the authentication process is a captive portal. From the test results, captive portal can be used to handle user authentication using a username and password that is registered in the LDAP server. Using a proxy server can speed up the response time for web pages to load. Proxy server performance can be evaluated from the ratio of the percentage of hits. Load balancing can improve the hit ratio of each server, ranging between 20-30%.

Keyword: proxy, caching, transparent, authentication, RADIUS, captive portal, load balancing, IPTables, round robin, ratio of hit

1. Pendahuluan

1.1 Latar Belakang

Penyeimbangan beban yang dibantu *peering* dapat meningkatkan kinerja jaringan, karena *server proxy* akan dibantu melayani beban oleh beberapa *server proxy* yang lain. *Overload* beban atau *throughput*, bisa menimbulkan kemacetan (*bottle-neck*), yang mengganggu kenyamanan jaringan Internet itu sendiri.

Penggunaan open source yang digunakan pada perancangan ini dapat secara fleksibel mengikuti persyaratan sistem yang ingin diimplementasikan. Sistem penyeimbangan beban ini bisa didukung aplikasi autentikasi dengan interkoneksi ke *server* LDAP. Pengguna yang sudah terautentikasi akan dialihkan trafik HTTP-nya oleh *gateway* ke masing-masing *server proxy* yang telah ditentukan dalam penyeimbangan beban.

1.2 Tujuan

Tujuan dari penelitian ini adalah perancangan dan implementasi *server gateway captive portal* dan penyeimbangan beban Squid/LUSCA *proxy*.

1.3 Batasan Masalah

Adapun pembatasan masalah pada makalah ini adalah sebagai berikut :

1. Gateway yang digunakan merupakan sebuah server dengan sistem operasi Ubuntu Server 10.04 LTS. Layanan yang akan digunakan adalah FreeRADIUS, Coova Chilli, dan Enginx, sebagai aplikasi *captive portal*, VLAN, dan *bridge* untuk menjalankan layanan *router-on-a-stick*.
2. Hanya membahas penyeimbangan beban yang dilakukan oleh IPTables pada mode n^{th}. IPTables juga digunakan untuk mengalihkan trafik dari pengguna ke beberapa *server proxy* yang telah ditentukan.
3. *Server proxy* menggunakan sistem operasi Ubuntu Server 12.04 LTS dengan paket LUSCA, yang bisa melakukan *caching* konten dinamis seperti video dan konten Flash.
4. Pembuatan *multiple instance* dari *captive portal*, memerlukan beberapa *file* konfigurasi, untuk membedakan *port*, PID, direktori, dan IPTables yang digunakan.

2. Metode

Server gateway captive portal merupakan server yang menyediakan layanan RADIUS untuk autentikasi dalam penggunaan jaringan Internet. RADIUS memungkinkan autentikasi bisa dilakukan dengan mengambil data *username* dan *password* dari *server* LDAP yang telah dibuat sebelumnya. RADIUS hanya menyediakan fasilitas untuk autorisasi dan autentikasi saja, sehingga diperlukan CoovaChilli sebagai *walled-garden* untuk bisa mencegat dan mengarahkan pengguna yang ingin terhubung ke Internet ke sebuah halaman yang mengharuskan pengguna untuk *login* menggunakan username dan password yang terdaftar di *server* LDAP.

CoovaChilli juga menggunakan aplikasi *web captive portal* Enginx yang digunakan untuk mengirimkan parameter *access controller* yang terhubung ke *server* RADIUS. Parameter ini digunakan untuk mengirimkan dan menampilkan status dari pengguna melalui tampilan *web*. Enginx *website* akan mempermudah pengguna untuk melakukan *login* dan *logout* karena *interface web*-nya yang berbasis GUI, sehingga tinggal mengisi *field* yang disediakan.

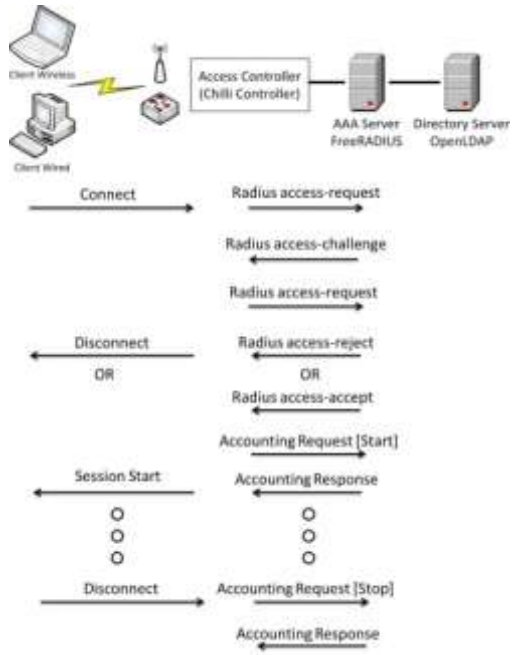
IPTables melakukan penyeimbangan beban dengan membagi paket secara bergantian ke masing-masing *server proxy* yang tersedia. *Server proxy* dirancang untuk menggunakan *hirarki sibling* yang bisa membantu proses pencarian konten yang pernah diakses di antara ketiga *proxy*. Konten yang sudah pernah disimpan sebagai *cache* di *proxy* yang diatur sebagai *sibling* bisa membuat UDP *hit* di *server proxy* yang lain, sehingga tidak perlu mencari konten ke *server* luar.

Perangkat *hub* tersedia pada masing-masing laboratorium, sedangkan perangkat yang digunakan pada perancangan kali ini terletak di Lab Komputer yaitu *switch* layer 2 seri 2960 Cisco, Mikrotik Routerboard RB750, sebuah *server gateway captive portal*, serta *hub* untuk ketiga *server proxy*. Perangkat di atas *switch* 2960 merupakan perangkat yang tersedia pada layer *access* sampai layer *distribution* UNDIP sehingga bukan merupakan wilayah administrasi.

Gambar 1. Topologi fisikal perancangan *server gateway captive portal* dan *proxy*

Gambar 2. Topologi logikal perancangan *server gateway captive portal* dan *proxy*

FreeRADIUS sebagai *server* RADIUS memiliki kemampuan standart untuk mengatur komunikasi antara *Network Access Server* (NAS) dengan *server* AAA. *Server* FreeRADIUS yang digunakan merupakan kesatuan dari server AAA dan juga NAS. Paket-paket data yang terlibat dalam komunikasi antara keduanya disebut sebagai paket RADIUS.

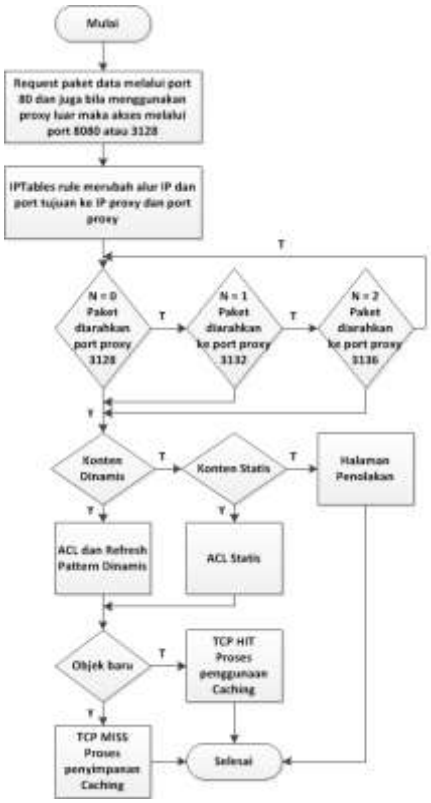


Gambar 3. Mekanisme paket RADIUS

IPTables mengarahkan semua permintaan HTTP pada *port* 80, 8080, 3128 dari pengguna pada masing-masing VLAN ke masing-masing *server proxy* secara *load balance*. IPTables akan membagi setiap paket secara merata menuju ketiga *server proxy* yang ada. Pengguna yang menggunakan *port* HTTP yang telah ditentukan di atas akan seolah-olah diwajibkan untuk meneruskan semua permintaannya ke *server proxy*.

LUSCA yang digunakan sebagai *head* membuat Squid bisa mendeteksi konten-konten dinamis seperti video. ACL khusus diperlukan untuk mendeteksi pola dari konten dinamis yang biasanya berisi *path* URL berdasarkan aturan *regex*, *refresh pattern*, dengan dibantu *script* PERL untuk melakukan fungsi *split* dan penggantian pola. Konten statis secara *default* bisa di-*caching* oleh Squid.

Konten yang sudah pernah di-*caching* dan belum kadaluarsa bisa membuat Squid memperoleh status TCP *hit*, yang artinya konten diambil dari *server proxy*, bukan dari *server* tujuan.



Gambar 4. Alur proses *proxy*

3. Hasil dan Analisa

yang akan diterapkan pada *server gateway captive portal* adalah FreeRADIUS sebagai NAS *client* dan *server* AAA yang terhubung ke *server* LDAP, CoovaChilli sebagai *walled-garden* dan Enginx *website* yang menyediakan *interface web* yang memudahkan *login*, selain itu ada paket VLAN yang memungkinkan penandaan VLAN pada *virtual interface*, paket *bridge* yang digunakan untuk memuat VLAN, serta juga paket IPTables yang akan berfungsi sebagai *gateway* sekaligus sebagai *load balancer*.

Sistem yang akan diterapkan pada *server proxy* adalah paket LUSCA serta squidclient untuk melakukan *monitoring* kinerja Squid. Masing-masing server akan diberi layanan Monitorix untuk memantau semua penggunaan *resource* CPU, memori, I/O, dan *bandwidth* dari *interface* yang ada..

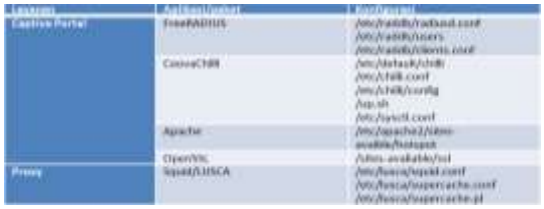
3.1 Implementasi dan Konfigurasi FreeRADIUS

Aplikasi FreeRADIUS akan memerlukan beberapa dukungan *library* seperti libldap2 dan libperl untuk berkomunikasi dengan LDAP dan juga mendukung bahasa PERL.

3.2 Implementasi dan Konfigurasi CoovaChilli

Implementasi CoovaChilli tidak memanfaatkan *repository* Ubuntu tapi memanfaatkan paket debian yang tersedia di *website* coova.org.

Tabel 1. Tabel konfigurasi pada masing-masing aplikasi



Percobaan akan dilakukan dengan menggunakan *web browser* untuk mengakses sebuah situs di Internet. Pengguna setelah itu akan langsung diarahkan ke halaman *login* Enginx *website*.

Halaman *login* Enginx *website* akan langsung ditampilkan setelah proses pemasangan sertifikat selesai seperti pada Gambar 4.4. Proses *login* akan dinyatakan berhasil setelah memasukkan *username* dan *password* yang terdaftar pada *server* LDAP dengan benar. Hubungan menuju jaringan Internet akan diizinkan, ketika proses *login* berhasil.

Gambar 5. Halaman *login captive portal*

3.3 Implementasi dan Konfigurasi VLAN

Mikrotik RB750 atau Cisco Catalyst 2960 bisa menjadi pilihan untuk menjadi *switch* layer 2 yang berguna dalam membuat keanggotan akses VLAN.

Interface ether1, ether2, dan ether4 pada Mikrotik akan menjadi vlan access untuk vlan 316, 318, dan 319. Interface ether3 pada Mikrotik akan menjadi *trunk port*. *Interface* fa0/1 pada Catalyst akan menjadi VLAN access untuk vlan 317 dan *interface* fa0/4 akan menjadi *trunk port*.

3.4 Implementasi Penerusan Paket dengan IPTables

IPTables memegang peranan penting untuk meneruskan paket dari *interface gateway* yang terkoneksi ke Internet, melakukan *forwarding* dan juga membuat VLAN bisa berkomunikasi satu dengan lainnya ataupun dengan jaringan lokal yang lain, serta juga melakukan penyeimbangan beban dan DNAT agar semua trafik HTTP dari VLAN bisa dibelokkan menuju ketiga *server proxy* yang ada. IPTables dalam menyeimbangkan permintaan menggunakan algoritma round robin yang akan membagi paket berdasarkan urutan paket datang ke masing-masing server dengan sistem antrian sirkuler.

3.5 Implementasi Squid/LUSCA *Proxy*

Implementasi Squid/LUSCA pada sistem operasi Ubuntu Server 12.04 dapat menggunakan paket dari *repository* yang ada, sehingga akan lebih mudah bila dibandingkan implementasi Squid/LUSCA pada Ubuntu Server 10.04. Ubuntu 12.04 akan menggunakan *instance* LUSCA secara *default* sehingga tidak perlu melakukan *patch* dan *build* dari *source*.

3.6 Hasil Pengujian *Server Gateway Captive Portal*

Pengujian pada server gateway captive portal dilakukan selama kurang lebih satu bulan. Server *gateway captive portal* merupakan komponen utama yang seharusnya tidak boleh *down*, karena bila *down*, pembagian koneksi Internet dari masing-masing laboratorium tidak berjalan. Pengujian kali ini tidak menggunakan *captive portal*. *Server gateway captive portal* hanya melakukan layanan bridge VLAN, serta juga melakukan layanan *routing* dan NAT menggunakan IPTables.

Throughput Internet merupakan gabungan dari aktifitas *downstream* dan upstream. Grafik berwarna hijau merupakan aktifitas *downstream* yang meliputi aktifitas *download* dari pengguna dan grafik berwarna biru merupakan aktifitas *upstream* ketika melakukan permintaan ke *server* tujuan.

Gambar 6. Tampilan *throughput* total Internet pada *server gateway captive portal*

Informasi yang bisa diperoleh dari Gambar 12. adalah bahwa penggunaan trafik akan meningkat mulai dari pukul 12.00 siang, karena banyaknya mahasiswa melakukan aktifitas *download* pada jam kerja. Penggunaan *bandwidth* masih berlangsung hingga malam hari dan akan mulai menurun pada sekitar pukul 06.00 pagi. Penggunaan *bandwidth* maksimal rata-rata adalah

sekitar 3,6 MB dengan penggunaan rata-rata sekitar 1,5 MB.

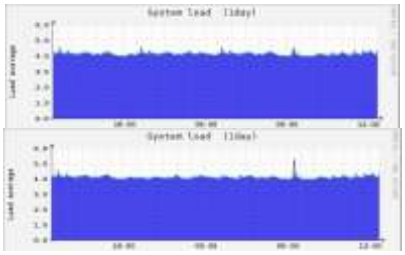
3.7 Hasil Pengujian *Server Proxy*

Pengujian pada *server proxy* juga dilakukan selama kurang lebih satu bulan. *Server proxy* merupakan komponen utama yang akan melayani permintaan trafik HTTP. Pada lingkungan laboratorium Teknik Elektro UNDIP dicoba diterapkan tiga *server proxy* internal untuk membantu *server proxy* utama UNDIP. Ketiga *server proxy* tersebut, sebelum dilakukan penyeimbangan beban, dicoba untuk memberikan layanan untuk masing-masing laboratorium yang berbeda. Hal ini berdasarkan pertimbangan bahwa pemakaian *satu server proxy* saja akan memberatkan *server* tersebut, bila harus menangani permintaan seluruh laboratorium yang ada. Hal ini juga memungkinkan terjadinya penurunan kinerja pada salah satu *server proxy* yang memiliki beban trafik yang lebih besar karena aktifitas pemakaian lebih banyak, sedangkan *server* yang lain tidak dimanfaatkan secara optimal.

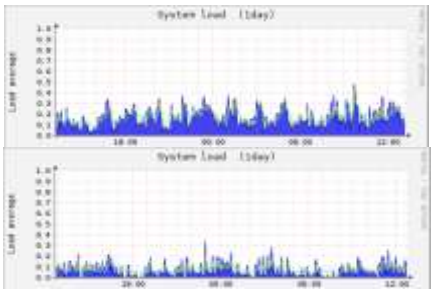
Ketiga *server proxy*, setelah dilakukan penyeimbangan beban, akan bersama-sama menangani permintaan dari seluruh laboratorium yang ada dengan algoritma penjadwalan *round robin*. Algoritma *round robin* mengarahkan koneksi jaringan ke beberapa *server* dengan cara *round robin*, dengan memperlakukan semua *server proxy* yang tersedia secara sama, terlepas dari jumlah koneksi atau waktu responnya. Algoritma *round robin* yang dipakai di sini berbeda dengan sistem *round robin* yang bekerja pada DNS. *Round robin* pada DNS akan menerjemahkan sebuah domain ke beberapa alamat IP yang berbeda, algoritma penjadwalan didasarkan pada setiap *host*, *caching* DNS akan menyembunyikan algoritma ini karena akan selalu berusaha memakai *host* yang terakhir kali diakses bila *cache* belum hilang, sehingga menyebabkan beban tidak seimbang menuju ke beberapa *server* [19]. Algoritma *round robin* yang dipakai pada IPTables didasarkan pada setiap koneksi, sehingga koneksi akan dibagi ke beberapa *server* yang tersedia berdasarkan urutan antriannya.

a. Utilisasi CPU

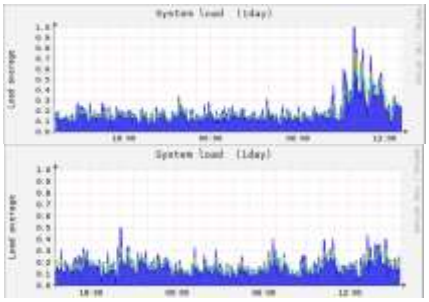
Gambar 7. menunjukkan utilisasi CPU pada *server* proxy1. Gambar 8. menunjukkan utilisasi CPU pada *server* proxy2. Gambar 9. menunjukkan utilisasi CPU pada *server* proxy3.



Gambar 7. Utilisasi CPU pada *server* proxy1 sebelum dan sesudah *load balancing*



Gambar 8. Utilisasi CPU pada *server* proxy2 sebelum dan sesudah *load balancing*



Gambar 9. Utilisasi CPU pada *server* proxy3 sebelum dan sesudah *load balancing*

Informasi yang bisa diperoleh dari Gambar 7. adalah bahwa utilisasi CPU pada *server* proxy1 terlihat lebih besar dibandingkan ketiga *server proxy* yang ada karena *server* proxy1 menggunakan spesifikasi Pentium III sedangkan yang lain Pentium Dual Core. Nilai utilisasi CPU *server* proxy1 sekitar 4 persen, sedangkan untuk *server* proxy2 dan proxy3 sekitar 0,2-0,3 persen, dari total 100 persen. Pada *server* proxy2 dan proxy3 terjadi sedikit penurunan utilisasi CPU setelah diseimbangkan bebannya, namun tidak signifikan. Dari hasil sebelum dan sesudah penyeimbangan beban, bisa disimpulkan bahwa layanan *proxy* tidak memerlukan utilisasi CPU yang besar.

b. Penggunaan Memori (RAM)

Grafik berwarna merah menunjukkan memori yang dipakai oleh aplikasi yang berjalan. Grafik berwarna hijau menunjukkan *buffer* dan *cache* yang digunakan oleh sistem operasi. Gambar 4.33 menunjukkan penggunaan

RAM pada *server* proxy1. Gambar 4.34 menunjukkan penggunaan RAM pada *server* proxy2. Gambar 4.35 menunjukkan penggunaan RAM pada *server* proxy3.

Gambar 10. Penggunaan memori pada *server* proxy1 sebelum dan sesudah *load balancing*

Gambar 11. Penggunaan memori pada *server* proxy2 sebelum dan sesudah *load balancing*

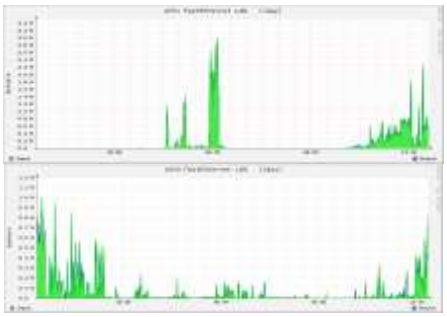
Gambar 12. Penggunaan memori pada *server* proxy3 sebelum dan sesudah *load balancing*

Informasi yang diperoleh dari gambar di atas adalah bahwa penggunaan RAM untuk layanan *proxy* akan cukup besar bila dibandingkan dengan *server gateway captive portal*. Perbedaan penggunaan memori sebelum dan setelah penyeimbangan beban terlihat tidak signifikan, karena selalu mendekati penggunaan sebesar 80-90%. Linux selalu berusaha membuat semua memori bekerja, seperti untuk *buffer* dan *cache*, sehingga penggunaan memori tidak dapat menunjukkan kinerja dari penyeimbangan beban [13].

c. Penggunaan *Throughput*

Throughput di sini menunjukkan besar konten yang masuk atau diakses oleh *server proxy*. Gambar 4.36 akan menunjukkan penggunaan *throughput* pada *server* proxy1. Gambar 4.37 akan menunjukkan penggunaan *throughput* pada *server* proxy2. Gambar 4.38 akan menunjukkan penggunaan *throughput* pada *server* proxy3.

Gambar 13. Penggunaan *throughput* total pada *server* proxy1 sebelum dan setelah *load balancing*



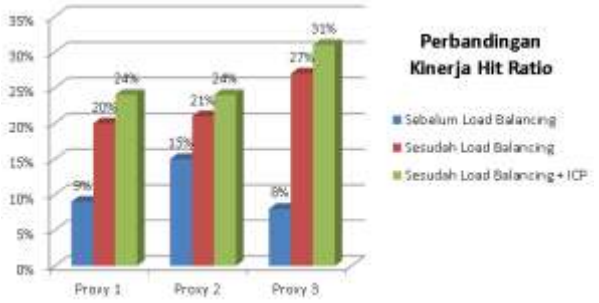
Gambar 14. Penggunaan *throughput* total pada *server* proxy2 sebelum dan setelah *load balancing*

Gambar 15. Penggunaan *throughput* total pada *server* proxy3 sebelum dan setelah *load balancing*

Informasi yang bisa diperoleh dari ketiga gambar di atas adalah bahwa sebelum dilakukan penyeimbangan beban, *server proxy* hanya mengalami *throughput* puncak pada jam-jam tertentu sesuai dengan aktifitas yang terjadi pada masing-masing laboratorium. Setelah dilakukan penyeimbangan beban, *server proxy* terlihat memiliki *throughput* pada setiap jam-nya walaupun persebarannya belum merata, namun menunjukkan bahwa *server proxy* bekerja secara bersamaan.

d. Hit Ratio

Salah satu parameter yang digunakan untuk mengukur kinerja *server proxy* adalah rasio *hit*. Rasio *hit* yang semakin tinggi menunjukkan bahwa *server proxy* menyediakan lebih banyak konten untuk pengguna daripada mengambil langsung dari *server* sesungguhnya. Rasio *hit* dapat diperoleh dari aplikasi squidclient. Aplikasi squidclient akan memberikan informasi yang detail tentang penggunaan *cache* pada *proxy*. Rasio *hit* diperoleh dengan membagi jumlah *Hits* dengan jumlah *Request*, kemudian nilainya dikalikan dengan 100%.



Gambar 16. Perbandingan kinerja *hit ratio* pada ketiga *server proxy*

3.8 Perbandingan Sebelum dan Sesudah Pemakaian *Server Proxy*

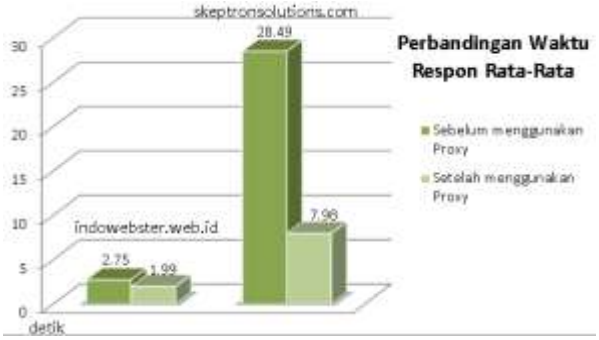
Parameter perbandingan yang dipakai untuk menguji kecepatan dan kualitas penggunaan *proxy* adalah dari segi *response time* dan juga segi *packet loss* saat *streaming.*

a. Perbandingan *Response Time*

Response time yang dimaksud disini merupakan waktu yang dibutuhkan untuk bisa memuat halaman *web* dengan kontennya secara penuh.

Pengujian *response time* di sini akan menggunakan aplikasi berbasis *web* yaitu WebWait, karena aplikasi seperti httperf tidak bisa memuat konten secara keseluruhan. *Proxy* akan menyimpan konten berupa gambar, skrip, atau *banner* yang sering diakses, oleh karena itu pada pengujian, waktu memuat gambar, skrip, atau banner juga harus diperhitungkan. WebWait dapat mengukur kecepatan pemuatan halaman dari *web* lain dengan memperhatikan aspek *rendering* dari berbagai gambar, *stylesheets*, dan juga JavaScript. Waktu respon yang diperoleh adalah waktu sebenarnya karena akan melalukan proses pemuatan halaman langsung dari *browser* pengguna, sehingga tidak menggunakan waktu simulasi.

Gambar 4.48 berikut menunjukkan grafik perbandingan pengujian waktu respon rata-rata secara lengkap, setelah dan sebelum menggunakan *proxy*.

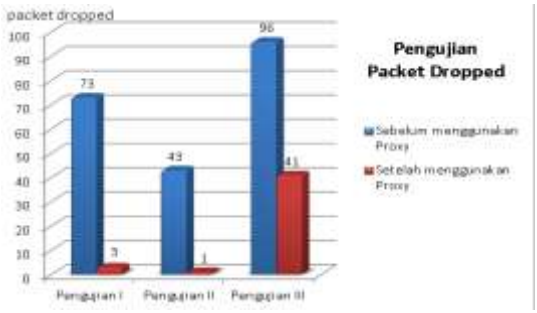


Gambar 17. Grafik perbandingan waktu respon rata-rata lengkap

b. Perbandingan *Packet Loss* saat *Streaming*

Packet loss atau *dropped* pada saat streaming berhubungan dengan *Quality of Service* (QoS). QoS menunjukkan kemampuan sebuah jaringan untuk menyediakan layanan yang lebih baik dengan menggunakan beberapa teknologi. QoS bisa disebut sebagai kumpulan teknologi yang mengijinkan aplikasi untuk melakukan permintaan dan penerimaan layanan pada dengan memperhatikan kapasitas *throughput*, variasi *latency* dari *jitter*, dan juga waktu tunda. QoS yang diuji disini berkaitan dengan *packet dropped* yang terjadi saat *streaming* situs *video* seperti youtube. Perbaikan QoS ditandai dengan peningkatan karateristik *loss*. Pada saat *streaming* data akan dikirim terus menerus tanpa memperhatikan data tersebut benar-benar sampai ke tujuan atau tidak (*connectionless*). Situs youtube menyediakan fasilitas informasi *streaming video* secara *real time*. Informasi *packet dropped* mengindikasikan adanya proses *buffering* yang ditunjukkan dengan "*spinning circle*". Peningkatan QoS ditunjukkan dengan penurunan nilai *packet dropped*, yang menunjukkan bahwa lebih banyak paket yang sampai ke tujuan. Berikut merupakan hasil pengujian situs youtube sebelum dan sesudah memakai *proxy*.

Gambar 4.55 berikut menunjukkan grafik perbandingan pengujian *packet dropped* secara lengkap, setelah dan sebelum menggunakan *proxy*.



Gambar 18. Grafik perbandingan *packet dropped* lengkap

3.9 Kelemahan Sistem

Kelemahan yang terjadi pada pengujian ini adalah *proxy* pada mode transparan yang diseimbangkan bebannya oleh IPTables tidak bisa melakukan blok situs tertentu dan *filtering* pada batasan ukuran *download* secara sempurna, bila dilakukan *refresh* dari pengguna. Hal ini disebabkan oleh kemampuan IPTables dalam menangani proses penyeimbangan beban. Ada paket yang dilemparkan ke *server gateway captive portal* secara lebih cepat daripada memprosesnya terlebih dahulu untuk melalui ACL yang dibuat pada *server proxy*.

4. Kesimpulan

Dari hasil impelentasi dan pengujian dapat disimpulkan bahwa:

1. Penggunaan *captive portal* yang terkoneksi ke *server* LDAP bisa menangani proses autentikasi yang tidak bisa dilakukan oleh *proxy* dalam mode transparan.
2. Penggunaan *captive portal* dapat membatasi akses *Internet* hanya untuk pengguna yang terdaftar di *server* LDAP.
3. Layanan *proxy* tidak memerlukan penggunaan utilisasi CPU yang besar karena untuk Pentium III hanya membutuhkan sekitar 4% dan untuk Pentium Dual Core hanya membutuhkan 0,2-0,3%.
4. Penurunan utilisasi CPU setelah diterapkan sistem penyeimbangan beban tidak signifikan, sehingga tidak bisa dijadikan tolok ukur kinerja *proxy*.
5. Penggunaan memori sebelum dan setelah diterapkan sistem penyeimbangan beban berkisar antara 80-90%. Perubahan yang terjadi terlihat tidak signifikan, karena Linux berusaha untuk selalu memakai memori yang ada, sehingga tidak bisa dijadikan tolok ukur kinerja *proxy*.
6. Persentase rasio *hit* meningkat dengan digunakannya sistem penyeimbangan beban, sampai berkisar 20-30%. Sistem penyeimbang beban bisa membagi beban berdasarkan urutan, sehingga membuat *proxy* bekerja bersama-sama untuk meningkatkan kemungkinan rasio *hit*.
7. Penggunaan *proxy* dapat mempercepat waktu *response time* karena *proxy* membantu dalam menyediakan konten-konten berupa gambar dan iklan.
8. Penggunaan *proxy* dapat memperbaiki resiko paket yang hilang pada aktivitas *streaming*.
9. *Proxy* pada mode transparan yang diseimbangkan bebannya oleh IPTables tidak bisa melakukan blok situs tertentu dan *filtering* pada batasan ukuran *download* secara sempurna, bila dilakukan *refresh* dari pengguna.

Referensi

[1]. Anonymous. *CoovaChilli*. "http://www.coova.org/CoovaChilli". 2010.

[2]. Carter, Gerald. *LDAP System Administration*. O'Reilly Media, Inc. 2003.

[3]. Chatel, M. *Classical versus Transparent IP Proxies*. Network Working Group RFC:1919. 1996.

[4]. Chadd, Adrian. *LUSCA Web Proxy Cache*. "http://www.lusca.org/Home". 2010.

[5]. David. *CoovaChilli JSON Interface*. "http://coova.org/CoovaChilli/JSON". 2008.

[6]. Hassell, Jonathan. *RADIUS*. O'Reilly. USA. 2002.

[7]. Muliawati, dkk. Analisa dan Perancangan Sistem Load Balancing Proxy Server (Studi Kasus : Universitas Bina Nusantara). Binus. Jakarta. 2004.

[8]. Nakhjiri. *AAA based Keying for Wireless Handovers: Problem Statement*. Network Working Group. 2006.

[9]. Novell. *SUSE Linux Administration Guide*. "http://www.novell.com/documentation/suse91/suselinux-adminguide/html/". 2012.

[10]. Pearson, Oskar. *Squid A User's Guide*. Qualica Technologies (Pty) Ltd. South Africa. 2000.

[11]. Ranch, David A. *Linux IP Masquerade HOWTO*. "http://www.tldp.org/HOWTO/IP-Masquerade-HOWTO/". 2005.

[12]. Technet Microsoft. *RADIUS Protocol*. "http://technet.microsoft.com/en-us/library/cc781821(WS.10).aspx". 2005.

[13]. Tranter, J. *Tips for Optimizing Linux Memory Usage*. "http://www.linuxjournal.com/article/2770". 2008.

[14]. Saini, Kulbir. *Squid Proxy Server 3.1 Beginners Guide*. Packt Publishing. Birmingham. 2011.

[15]. Sukoco, Heru. Implementasi dan Evaluasi Kinerja Load Balancing pada Server-Server Proxy di IPB. IPB. Bogor. 2008.

[16]. Syarif, Muhar. *Implementasi IPTables sebagai filtering Firewall*. Universitas Sriwijaya. 2008.

[17]. Van der Walt, Dick. *FreeRADIUS Beginner's Guide*. Packt Publishing. Birmingham. 2011.

[18]. Visolve Squid Team. *Transparent Cache Implementation Using Squid*. 2006.

[19]. Wensong, Z. *Linux Virtual Server: Linux Server Clusters for Scalable Network Services*. Free Software Symposium. 2002.

[20]. Wessels, Duane and K. Claffy. *Application of Internet Cache Protocol (ICP), version 2*. Network Working Group RFC:2187. 1997.

[21]. Wessels, Duane. *Configuring Hierarchical Squid Caches*. "http://old.squid-cache.org/Doc/Hierarchy-Tutorial/". 1997.